TP1

Programmation Objet Java

1 Objectifs

L'objectif de ce premier TP est de pratiquer les concepts basiques de la programmation objet. Pour cela, nous travaillerons dans le contexte présenté dans les deux premières PC.

2 Contexte

Le contexte des PC1-2 et du TP1 du module de programmation objet va être commun. Il s'agit de développer et étendre un logiciel simple de gestion de stock. Celui-ci sera à terme constitué de trois classes : **`Produit`**, **`Stock`** et **`Producteur`**.

Un producteur fabrique des produits et les range dans un stock unique. A un moment donné le producteur ne connaît que le produit qu'il est en train de créer. Un stock contient de nombreux produits (qui pourraient être produits par plusieurs producteurs). Chaque produit n'est produit que par un producteur et n'est entreposé que dans un seul stock.

Les classes **`Produit`** et **`Stock`** vues en PC vous sont fournies, elles sont accessibles sous :

~lib-src/FIP/INF111/TP1

3 API Java

L'API ("Application Programming Interface") Java est consultable via un navigateur à l'adresse : **`http://docs.oracle.com/javase/6/docs/api/index.html`**

- Recherchez la classe **`String`**
- Trouvez dans **`Method Summary`** la spécification des méthodes **`indexOf`** de la classe **`String`**

4 Lancement de jGRASP

jGRASP est un **E**nvironnement de **D**éveloppement **I**ntégré qui comprend, entre autres :

- Un éditeur de texte capable de s'adapter à la syntaxe de plusieurs langages
- Des liens vers des compilateurs
- Des commandes d'exécution [et de débogage pour Java]

Après vous être connecté, lancez **jGRASP** :

`Applications -> Programmation -> jGRASP`

En utilisant les icônes de la fenêtre gestion des fichiers, nous vous proposons :

- de créer un répertoire **INF111** qui contiendra tous les codes associés à ce module. (icône ).
- de vous placer dans ce répertoire.

Recopiez chez vous le répertoire dans lequel se trouvent les classes **Produit** et **Stock** en vous plaçant dans une fenêtre "terminal" dans le répertoire **INF111** créé avec jGRASP et en tapant la commande suivante :

```
cp -r  ~lib-src/FIP/INF111/TP1 .
```

Ouvrez la classe **Produit.java**

Testez quelques possibilités de jGRASP : indentation automatique, numérotation des lignes, … : (menu principal : ).

Compilez. (menu principal : **Build -> Compile** ou l'icône croix verte ). Un fichier **Produit.class** est alors créé.

Remarquez que **jGRASP** vous montre les commandes qu'il exécute dans la fenêtre compilation.

Lancez l'exécution du main de la classe Produit. (menu principal : Build -> Run ou l'icône petit bonhomme )

Remarquez que **jGRASP** vous montre l'exécution du programme dans la fenêtre exécution.

5 Utilisation de javadoc

La commande **javadoc** permet de créer la documentation d'un paquetage, d'une classe à la manière de la documentation des API de java que vous avez en ligne.

Dans une fenêtre "terminal" placez vous dans le répertoire **TP1** et créez un répertoire **doc** qui contiendra les fichiers html de la doc de vos classes

```
mkdir doc
```

Tapez la commande **javadoc** pour créer la documentation de vos deux classes

```
javadoc  –d  doc   *.java
```

L'option –d indique à javadoc où ranger la documentation. Avec un navigateur ouvrez le fichier index.html créé et comparez avec le contenu des commentaires des fichiers Produit.java et Stock.java

Il est possible de générer automatiquement la documentation en utilisant l'icône approprié sous jgrasp.

Pour en savoir plus, faîtes une recherche web avec les mots clefs `javadoc tag`.

6 Travail à réaliser

La classe **Producteur** crée des objets de type **Produit** et les range dans un objet **Stock**. Un **Producteur** possède un nom (**name**) qui lui est attribué à sa création, il connaît aussi à sa création l'objet de type **Stock** (**stock**) où il doit ranger ses produits.

Dans un premier temps, les méthodes ne gèrent pas les erreurs d'ajout d'un **`Produit`** dans un **`Stock`** plein ou de retrait d'un **`Produit`** d'un **`Stock`** vide.

Vous devez réaliser la classe **`Producteur`** dont les constructeurs et méthodes sont résumés ci-dessous.

- Constructeurs :

  un **constructeur** avec comme paramètres son nom et le stock qu'il utilise

- Méthodes d'instances :

  `getName()` : donne le nom du **`Producteur`**

  `setName(String)` : change le nom du **`Producteur`**

  `getStock()` : donne le **`Stock`** utilisé par le **`Producteur`**

  `produce(String)` crée un produit du nom du paramètre et le stocke dans le stock.

  `toString()` : rend une chaîne de caractères décrivant le **`Producteur`**

- Tests

  Une méthode **`main`** qui teste la classe **`Producteur`**

7 Modifications possibles

Comment faire pour qu'un **`Produit`** "connaisse" son **`Producteur`**? Faîtes les modifications nécessaires dans la classe **`Produit`**. Modifiez aussi la méthode **`produce(String)`** de la classe **`Producteur`** de façon à créer un **`Produit`** qui "connaisse" son **`Producteur`**.

Comment faire pour que la méthode `toString` de la classe **`Produit`**, affiche non seulement le nom et le numéro du produit, mais aussi le nom de son **`Producteur`** ? Faîtes les modifications nécessaires.

Testez vos modifications.